Hướng dẫn lập trình Timer với STM32

0

By embedded | Published: January 17, 2015

I. Giới thiệu cơ bản về Timer trong STM32

– Timer trong STM32 có rất nhiều chức năng chẳng hạn như bộ đếm counter, PWM, input capture ngoài ra còn một số chức năng đặt biệt để điều khiển động cơ như encoder, hall sensors.
– Timer là bộ định thời có thể sử dụng để tạo ra thời gian cơ bản dựa trên các thông số: clock, prescaler, autoreload, repetition counter. Timer của STM32 là timer 16 bits có thể tạo ra các sự kiện trong khoảng thời gian từ nano giây tới vài phút gọi là UEV(update event).

- **Các thành phần cơ bản của một timer bao gồm:**

TIM_CLK: clock cung cấp cho timer.

PSC (prescaler): là thanh ghi 16bits làm bộ chia cho timer, có thể chia từ 1 tới 65535

ARR (auto-reload register): là giá trị đếm của timer (16bits hoặc 32bits).

RCR (repetition counter register): giá trị đếm lặp lại 16bits.

Giá trị UEV được tính theo công thức sau:

UEV = TIM_CLK/((PSC + 1)*(ARR + 1)*(RCR + 1))

Ví dụ: TIM_CLK = 72MHz, PSC = 1, ARR = 65535, RCR = 0.

UEV = 72000000/((1 + 1)*(65535 + 1)*(1)) = 549.3 Hz

Với giá trị ARR (auto-reload) được cung cấp thì bộ định thời (timer) thực hiện các chế độ đếm khác nhau: đếm lên, đếm xuống hoặc kết hợp cả 2. Khi thực hiện đếm lên thì giá trị bộ đếm bắt đầu từ 0 và đếm tới giá trị ARR-1 thì báo tràn. Khi thực hiện đếm xuống thì bộ đếm bắt đầu từ giá trị ARR đếm xuống 1 thì báo tràn.

- **Các chế độ hoạt động của Timer:**

– Input capture
– Output compare
– PWM generation (Edge- and Center-aligned modes)
– One-pulse mode output

II. Định nghĩa struct cấu hình cho timer cơ bản:

```
typedef struct
{
  uint16_t TIM_Prescaler;        /*!< Specifies the prescaler value used to divide the TI
```

Categories

Embedded Firmware
Embedded Linux
Embedded Training
Knowlege MCU
programming language
Project
Qt/QML C++
Raspberry pi

Find us on facebook

- Facebook Members WordPress Plugin

Recent Posts

Basic Embedded Linux Training_khóa 2019
Hướng dẫn build yocto and qt5 cho udoo neo

```
                                This parameter can be a number between 0x0000 and 0

  uint16_t TIM_CounterMode;       /*!< Specifies the counter mode.
                                This parameter can be a value of @ref TIM_Counter_M

  uint32_t TIM_Period;            /*!< Specifies the period value to be loaded into the ac
                                Auto-Reload Register at the next update event.
                                This parameter must be a number between 0x0000 and

  uint16_t TIM_ClockDivision;     /*!< Specifies the clock division.
                                   This parameter can be a value of @ref TIM_Clock_Divi

  uint8_t TIM_RepetitionCounter;  /*!< Specifies the repetition counter value. Each time t
                                reaches zero, an update event is generated and coun
                                from the RCR value (N).
                                This means in PWM mode that (N+1) corresponds to:
                                   - the number of PWM periods in edge-aligned mode
                                   - the number of half PWM period in center-aligne
                                This parameter must be a number between 0x00 and 0x
                                @note This parameter is valid only for TIM1 and TIM
} TIM_TimeBaseInitTypeDef;
```

*** Phân tích cấu trúc :**

– TIM_Prescaler : Tham số TIM_Prescaler hiểu đơn giản như một bộ chia tần số.

Ví dụ STM32F4 Tần số cao nhất mà clock timer 4 đạt được là 84Mhz sau khi qua bộ chia này sẽ ra tần số clock timer(Fc_timer). Và ví dụ chọn Fc_timer = 1Mhz <=> Fc_timer = 84000000/84. Và TIM_Prescaler có công thức là: ((SystemCoreClock/2)/1000000)-1 = 83 , do hệ đếm bắt đầu từ 0 chứ không phải là 1 như chúng ta vẫn hay dùng để đếm số, bắt đầu đếm từ 0 -> 83 sẽ là 84 giá trị.

TIM_Prescaler = ((SystemCoreClock/n)/Fc_timer)-1

* Notes : Tùy vào timer nào mà chỉ số chia n sẽ khác nhau. Ví dụ trong stm32f4 gồm có những timer và hệ số chia khác nhau như hình bên dưới :

TIMER	TYPE	RESOLUTION	PRESCALER	CHANNELS	MAX INTERFACE CLOCK	MAX TIMER CLOCK*	APB
TIM1, TIM8	Advanced	16bit	16bit	4	SysClk/2	SysClk	2
TIM2, TIM5	General purpose	32bit	16bit	4	SysClk/4	SysClk, SysClk/2	1
TIM3, TIM4	General purpose	16bit	16bit	4	SysClk/4	SysClk, SysClk/2	1
TIM9	General purpose	16bit	16bit	2	SysClk/2	SysClk	2
TIM10, TIM11	General purpose	16bit	16bit	1	SysClk/2	SysClk	2
TIM12	General purpose	16bit	16bit	2	SysClk/4	SysClk, SysClk/2	1
TIM13, TIM14	General purpose	16bit	16bit	1	SysClk/4	SysClk, SysClk/2	1
TIM6, TIM7	Basic	16bit	16bit	0	SysClk/4	SysClk, SysClk/2	1

– TIM_CounterMode : Thiết lập mode cho timer là đếm lên hay đếm xuống. Nếu chọn mode đếm tăng có nghĩa là mỗi xung nhịp timer, bộ đếm counter sẽ tự tăng lên một giá trị theo chiều dương cho đến khi nào bằng giá trị period sẽ đếm lại từ đầu, người ta thường gọi trường hợp này là tràn bộ đếm.

– TIM_Period : Period có nghĩa là chu kỳ của timer (không phải là chu kỳ của 1 xung clock timer). Ví dụ một chu kỳ gồm 1000 xung clock mà mỗi xung clock = 1us ta sẽ được period là 1ms. Tôi trừ đi cho 1 là vì hệ đếm bắt đầu từ 0 như đã giải thích bên trên..

- Notes : Khi cấu hình sử dụng timer ta cần quan tâm đến 3 yếu tố chính đó là :

+ Đếm với xung clock timer là bao nhiêu (Fc_timer – xác định qua TIM_Prescaler).

board

Basic Embedded Linux Training_khóa 2018

Hướng dẫn build linux với yocto project cho board beaglebone black, linux kernel 4.5.7

Hướng dẫn build yocto cho raspberry pi

Recent Comments

Đặng Vũ on Hướng dẫn build linux với yocto project cho board beaglebone black, linux kernel 4.5.7

Đặng Vũ on Hướng dẫn build linux với yocto project cho board beaglebone black, linux kernel 4.5.7

Hướng dẫn build linux với yocto project cho board beaglebone black, linux kernel 4.5.7 | Embedded System on Basic Embedded Linux Training_khóa 2016

Dat on Thu thập và xử lý dữ liệu với kit Raspberry pi qua module RFID_RC522

Pham Van Dong on Cài đặt và cấu hình môi trường phát triển Qt trên kit mini2440

+ Đếm lên hay đếm xuống (TIM_CounterMode).

+ Đếm đến bao nhiêu (TIM_Period).

III. Ví dụ :

1. Ví dụ cấu hình cho timer 3 trong stm32f1 tạo interrupt với 100ms

*** Phân tích : (lưu ý 1 giây = 1ms x 1000)**

– Tần số cao nhất mà timer 3 trong stm32f1 đạt được là 72Mhz.

– Yêu cầu cần timer 3 tạo interrupt mỗi 100ms.

– Giả sử cần counter của timer đếm 100 lần để được 100ms và phát interrupt thì ta có :

– > Tần sô Fc_timer3 = (100ms x 100 clock) x 10 = 10.000 clock/ 1 giây (10Khz).

Kết quả :

+ TIM_Prescaler = (72000000/10.000 – 1); //Fc_timer là 10khz

+ TIM_Period = 100 lần – 1 = 99

– Giả sử cần counter timer đếm 1000 lần để được 100ms và phát interrupt thì tương tự :

->Tần số Fc_timer3 = (100ms x 1000 clock) x 10 = 100.000 clock/ 1 giây (100Khz)

Kết quả :

+ TIM_Prescaler = (72000000/100.000 – 1); //Fc_timer là 100khz

+ TIM_Period = 1000 lần – 1 = 999

//Cấu hình các thông số cho timer3 với trường hợp period = 99:

TIM_TimeBaseStructure.TIM_Period = 99; // delay 10ms

TIM_TimeBaseStructure.TIM_Prescaler = (72000000/10000 – 1); //10khz

TIM_TimeBaseStructure.TIM_ClockDivision = 0;

TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;

TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);

Trong ví dụ này, chọn Fc_timer3 = 10000 (10khz) nghĩa là :

-> Tần số timer 10.000 xung / 1 giây <-> 10 xung/ 1ms

-> Tần số cho 10ms là : 100xung/10ms

2. Ví dụ cấu hình cho timer 4 trong stm32f4 tạo interrupt với mỗi 1ms

* Phân tích :

– Tần số cao nhất mà mà timer 4 trong stm32f4 đạt được là 84Mhz

– Yêu cầu cần timer 4 tạo interrupt mỗi 1ms.

– Giả sử cần counter của timer đếm 100 lần để được 1ms và phát interrupt thì ta có :

– >Tần sô Fc_timer4 = (1ms x 100 clock) x 1000 = 100.000 clock/ 1 giây (100Khz).

Kết quả :

+ TIM_Prescaler = (84000000/100.000 – 1); //Fc_timer là 100khz

+ TIM_Period = 100 lần – 1 = 99

– Giả sử cần counter timer đếm 1000 lần để được 1ms và phát interrupt thì tương tự :

-> Tần số Fc_timer4 = (1ms x 1000 clock) x 1000 = 1000.000 clock/ 1 giây (1Mhz)

Kết quả :

+ TIM_Prescaler = (72000000/1000.000 – 1); //Fc_timer là 1Mhz

+ TIM_Period = 1000 lần – 1 = 999 ;

//Cấu hình các thông số cho timer4 với trường hợp period = 999:

```
TIM_TimeBaseStructure.TIM_Prescaler = ((SystemCoreClock/2)/1000000)-1;    // f = 1Mhz
TIM_TimeBaseStructure.TIM_Period = 1000 – 1;
TIM_TimeBaseStructure.TIM_ClockDivision = 0;

TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
```

* Trường hợp timer 4 trong stm32f4, period = 999
– Yêu cầu đếm 1000 xung để được **interrupt 1ms** thì 1 xung sẽ tương ứng với 1us
– Hệ số chia TIM_Prescaler = 1.000.000 trong khi xung hệ thống cấp là 84.000.000 do đó mỗi xung của clock timer sẽ bằng 84 xung của system clock.

Leave a comment

embedded
http://hethongnhung.com , http://laptrinhnhung.com
View all posts by embedded →

No comments yet.

Leave a Comment

Name (Required)

Mail (will not be published) (Required)

Website

Submit Comment

 Site Admin Return to Top designed by linhdong24